COMP6200 Data Science

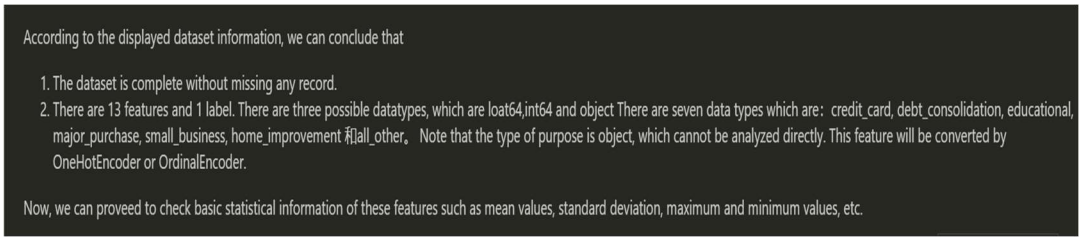
Critical Analysis Task

NAME: Muhammad Asad Imran Rafique

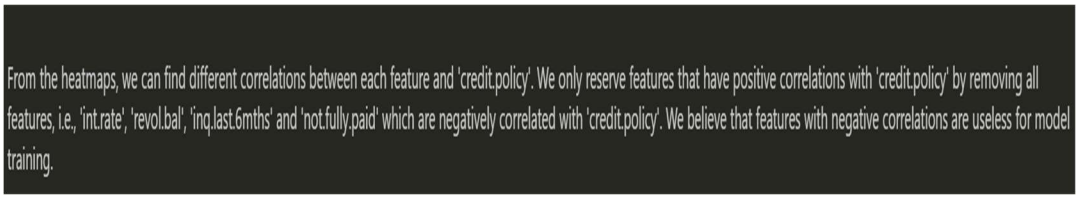
Student ID: 48065145

The issues I have located inside the given Jupyter Notebook are as follows:

1. The author checks for null values and replaces one null data point with the average of that feature. Then proceeds to statistical analysis and plotting correlation of the dataset without first encoding "purpose" using any encoding technique. The author even talks about encoding but does not perform it prior to analysis.



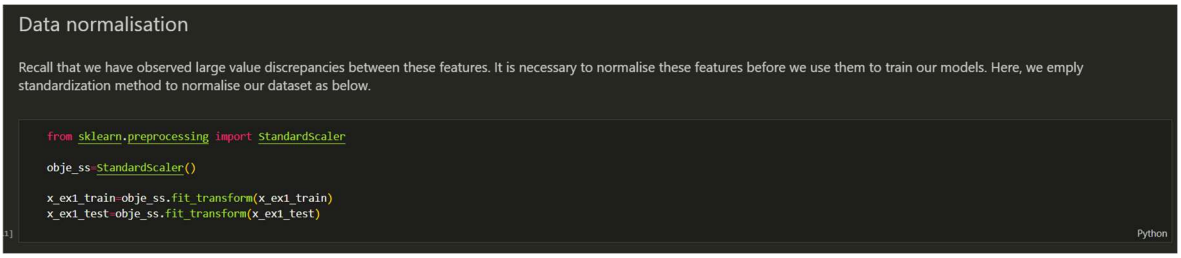
2. The author disregards the features with strong negative correlation as being irrelevant which is incorrect. Negative correlations are also useful for creating a prediction model.



3. Firstly, the best features should be extracted using RFE or similar techniques. Secondly, this is not the correct way to split data into training and testing data. The data will be split into training and testing data in the order of the original dataset. This could affect the model's performance due to bias. Instead, using the train_test_split function is a much better approach.

```
x_ex1 = New_Data.copy().drop(columns=['credit.policy', 'int.rate','revol.bal', 'inq.last.6mths', 'not.fully.paid' ])
y_ex1 = New_Data.copy()['credit.policy']
x_ex1_array = x_ex1.values
y_ex1_array = y_ex1.values
x_ex1_train = x_ex1_array[0:int((len(y_ex1_array)+1)*0.9),:]
x_ex1_test = x_ex1_array[int((len(y_ex1_array)+1)*0.9):,:]
y_ex1_train = y_ex1_array[0:int((len(y_ex1_array)+1)*0.9)]
y_ex1_test = y_ex1_array[int((len(y_ex1_array)+1)*0.9):]
```

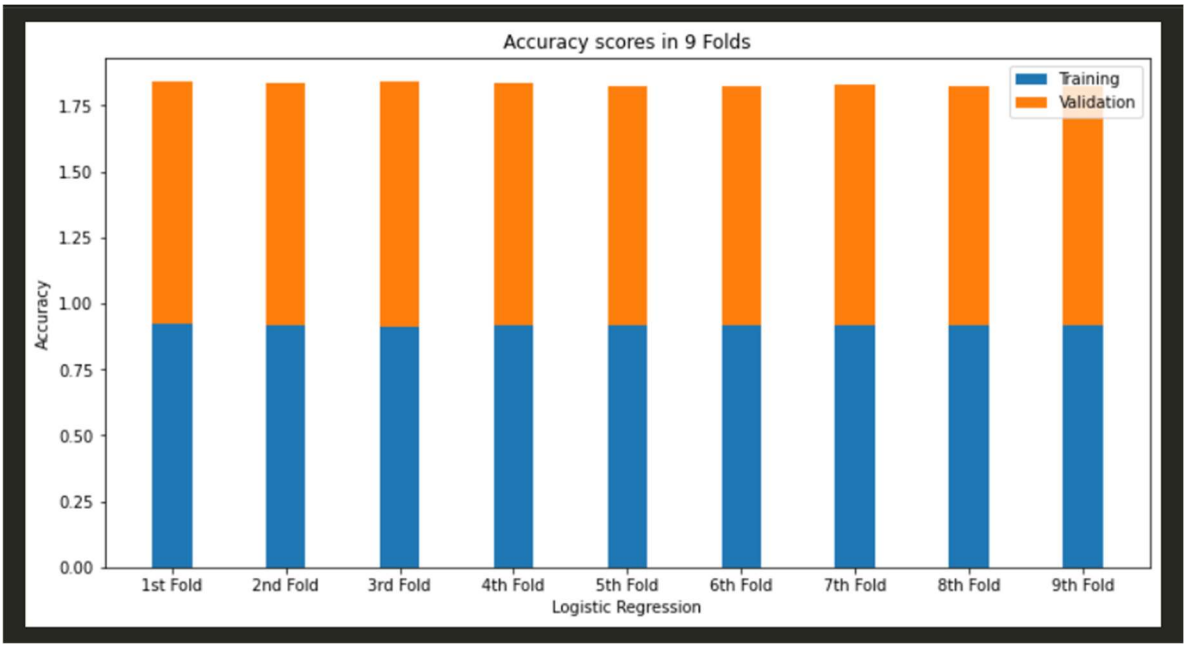
4. The data standardization technique is fine, but it could be improved. Instead of standardization, we should use normalization where the data of all fields fall in the range 0 to 1. Furthermore, normalization should be performed before splitting the dataset into training and testing data, instead of normalizing both training and testing data separately.



5. For cross validation, generally the entire X and Y data is passed rather than just the training data. The cross-validation function splits the data into training and testing differently for each fold. Currently, the training data is being further split into training and testing. This is not the normal way of model creation.

```
from sklearn.linear_model import LogisticRegression
model_log = LogisticRegression()
log_result = onbtain_cross_validition_results(model_log, x_ex1_train, y_ex1_train, _cv=9)
print(log_result)
```

6. The training and validation results for cross-validation of logistic regression model are plotted using stacked bar charts which is not visually helpful. A better approach should be to plot training and validation bars side by side for each fold. The conclusion drawn is meaningless unless we get a side-by-side comparison of both results. Also, the fact that accuracy score is going up to a Y value of 2 which is impossible.



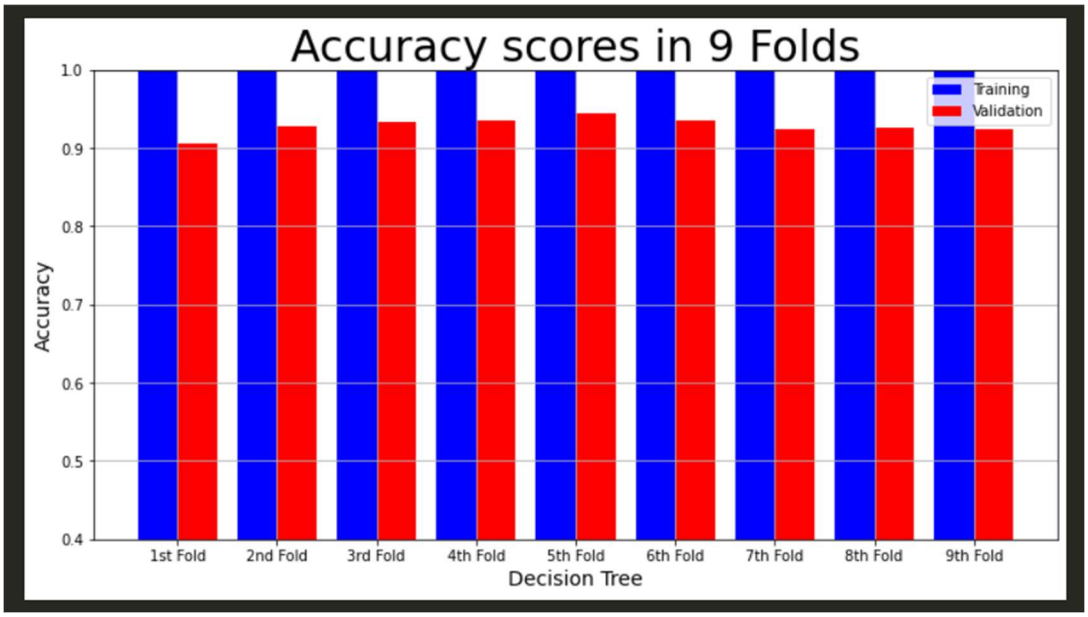
7. Like the logistic regression model, the cross-validation function for Decision Tree Classifier is also given only the training data instead of the entire X and Y data.

```
from sklearn.tree import DecisionTreeClassifier

model_tree = DecisionTreeClassifier()

tree_result = onbtain_cross_validition_results(model_tree, x_ex1_train, y_ex1_train, _cv=9)
print(tree_result)
```

8. The training and validation results for Decision Tree Classifier model are plotted side by side. This looks much better than the logistic regression results plots. However, the conclusion drawn regarding overfitting is subjective. The model performance is still over 90% which is certainly very good. Still, the best practice would be to get rid of any overfitting/underfitting.



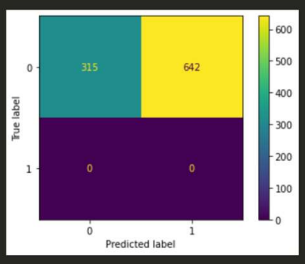
9. The hyper-parameter tuning can be improved by using all available choices of criterion for e.g. “log_loss” and increasing the range of the other parameters from (1,3) to maybe (1,10). The limited options bring bias into the testing which does not help the model’s accuracy. The tuned model has still catered for over-fitting, which was the goal, but it would be better to use all available options in tuning.

```
# We tune hyper-parameters as below, which could be pretty time consuming
from sklearn.model_selection import GridSearchCV
tree_para = {
    'criterion':['gini','entropy'],
    'max_depth':range(1,3),
    'min_samples_split':range(1,3),
    'min_samples_leaf':range(1,3),
        }
clf = GridSearchCV(model_tree, tree_para, cv=9)
clf.fit(x_ex1_train, y_ex1_train)
```

10. The confusion matrix shows the problem in testing data is that there is no data point where the true label is 1. This is clearly biased testing data since the original data contained over 80% data points with label 1.

```
DecisionTreeClassifier(max_depth=2)
Train success rate: % 95.59215868228745
Test success rate: % 32.9153605015674

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x27051032c10>
```



From the evaluation result on the test set, we can find that the prediction performance is very poor on the test set. The accuracy is only about 32% on the test set

11. The accuracy of the models can be improved if all the steps are performed correctly and in order.

- The first step is data cleaning, which is to get rid of any null values. This is done correctly in the Notebook.
- The next step should be to encode any object features into numerical values. This is done out of order in the Notebook.
- Then, statistical analysis should be performed, this would lead to the observation that the data fields have vastly different ranges and hence require normalization.
- After normalization, we must plot correlation heatmaps and select the best features for training the prediction models. Another approach can be RFE or similar functions for feature selection.
- Splitting the normalized data with the best selected features randomly into 90-10 split for optimal training and testing.
- Using the entire dataset for cross-validation to obtain training and validation results. The results can be compared side by side visually.
- Finally, using the prediction of models and using the confusion matrix or other metrics like accuracy, r^2, mse to guage the performance of the model.